CSE 142, Summer 2018
Programming Assignment #2: Space Needle (16 points)
Due Tuesday, October 8, 2018, 11:59 PM
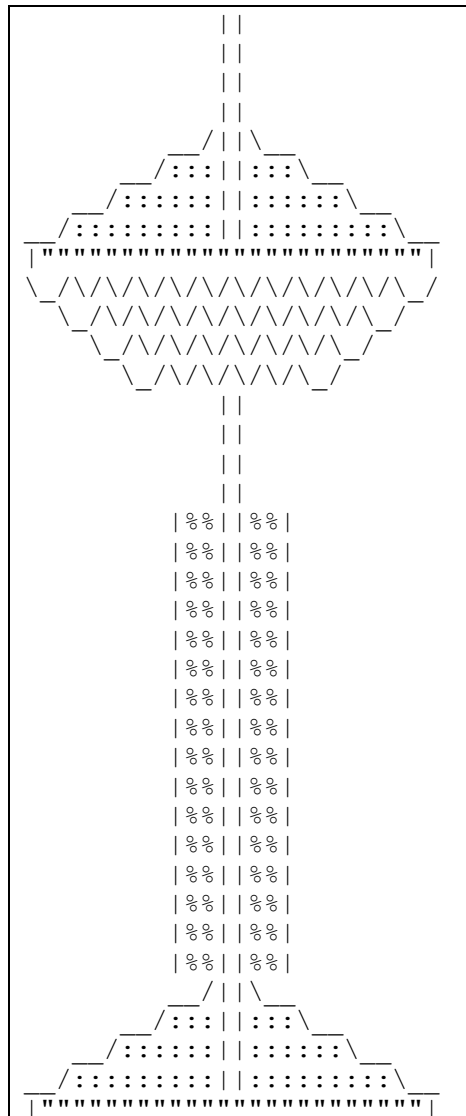
This assignment focuses on `for` loops, print/println statements, expressions, and constants. Turn in <u>two</u> Java files as described below.

Part A: ASCII Art (2 points):

The first part of your assignment is to write a program that produces any text art (sometimes called "ASCII art") picture you like. Write a Java class named `AsciiArt` in a file named `AsciiArt.java`. We will share everyone's art on the class website. Your program can produce any text picture you like, with the following restrictions and details:

- The picture should be your own creation, not an ASCII image you found on the Internet or elsewhere.
- The number of lines drawn should be at least 3 but no more than 200, and no more than 100 characters per line.
- The picture should not include hateful, offensive, or otherwise inappropriate images.
- The picture must not be substantially similar to your solution for Part B, consist entirely of reused Part B code, or be substantially similar to a related CSE142 assignments from a previous quarter.
- The code should use at least one `for` loop or static method but should not contain infinite loops.
- The code should not use material beyond Ch. 3 of the textbook.

If your Part A program compiles and runs successfully and meets the above constraints, it will receive the full 2 points. Part A will not be graded on "internal correctness".

```
                     | |
                     | |
                     | |
                     | |
                   __/ | | \__
                 _/:::||:::\_
               _/:::::||:::::\_
             _/:::::::||:::::::\_
           _/:::::::::||:::::::::\_
          |"""""""""""""""""""""""""|
          \_/\/\/\/\/\/\/\/\/\/\/\/_/
           \_/\/\/\/\/\/\/\/\/\/\/_/
            \_/\/\/\/\/\/\/\/\/\/_/
             \_/\/\/\/\/\/\/\/\/_/
              \_/\/\/\/\/\/\/\/_/
                     | |
                     | |
                     | |
                     | |
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                   |%%||%%|
                  __/ | | \__
                _/:::||:::\_
              _/:::::||:::::\_
            _/:::::::||:::::::\_
          _/:::::::::||:::::::::\_
         |"""""""""""""""""""""""""|
```

Part B: Space Needle (14 points):

The second part of your assignment is to produce a specific text figure that is supposed to look like Seattle's Space Needle. Turn in a class named `SpaceNeedle` in a file named `SpaceNeedle.java`. **You should exactly reproduce the format of the output at left. This includes having identical characters and spacing.**

One way to write a Java program to draw this figure would be to write a single `System.out.println` statement that prints each line of the figure. However, this solution would not receive full credit. A major part of this assignment is showing that you understand `for` loops.

In lines that have repeated patterns of characters that vary in number from line to line, represent the lines and character patterns using nested `for` loops. (See Chapter 2's case study.) It may help to write pseudocode and tables to understand the patterns, as described in the textbook and lecture.

Another significant component of this assignment is the task of generalizing the program using a single class constant that can be changed to adjust the size of the figure. See the next page for a description of this constant and how it should be used in your program.

The course web site will contain files that show you the expected output if your size constant is changed to various other values. You can use our Output Comparison Tool on the course web site to measure numbers of characters and to verify the correctness of your output for various values of the size constant.

Part B of this assignment will be graded both on "external correctness" (whether the program compiles and produces exactly the expected output) and "internal correctness" (whether your source code follows the style guidelines in this document).

(continued on back)

Style Guidelines (for Part B):

Use of `for` loops (nested as appropriate)

This program is intended to test your knowledge through Chapter 2, especially of nested `for` loops. If you like, you may also use the Java features from Chapter 3 such as parameters, although you are not required to do so and will receive no extra credit for doing so. You may not use any Java constructs beyond Chapter 3.

Use of static methods for structure and elimination of redundancy

Continue to use static methods to structure your solution in such a way that the methods match the structure of the output itself. Avoid significant redundancy; use methods so that no substantial groups of identical statements appear in your code. No `println` statements should appear in your `main` method. You do not need to use methods to capture redundancy in partial lines, such as the two groups of colons in the following line:

    __/::::::||::::::__

Source code aesthetics (commenting, indentation, spacing, identifier names)

You are required to properly indent your code. See the textbook for examples of proper indentation. No line of code should be more than 100 characters long.

Give meaningful names to methods and variables in your code. Follow Java's naming conventions for the format of `ClassNames`, `methodAndVariableNames`, and `CONSTANT_NAMES`.

Include a comment header at the beginning of your program with basic information and a description of the program. Also include a comment at the start of each method describing its behavior. Your comments should be your own words and should not directly copy this spec.

Class constant for figure's size

You should create one (and only one) class constant to represent the size of the parts of the figure. Use **4** as the default value of your constant to produce the figure shown above. Your figure <u>must</u> be based on that exact value to receive full credit.

On any given execution, your program will produce just one version of the figure. However, you should refer to the class constant throughout your code so that your program can produce a figure of a different size by simply changing the constant's value and recompiling. **Your program should scale the figure correctly for any constant value of 2 or greater.** Check the Output Comparison Tool to see what the output should look like for different sizes.
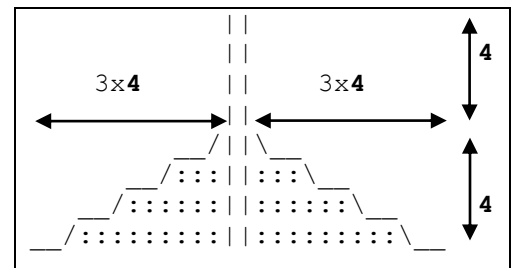
Please note that the height of the needle's midsection grows as the *square* of the figure's size. For example, in the default figure size of 4, the midsection is 16 lines tall. If the size were 7, the midsection would be 49 lines tall.

Development Strategy (How to Get Started):

This program is best completed in stages. We strongly recommend the following development strategy:

1. **Tables**: Examine the output and write tables to discover the patterns of repeated characters on each line.

2. **Code without constant**: Completely write the Java code to draw the Space Needle at its default size of 4.

3. **Code with constant**: Add a constant to your code so that the needle can scale to different sizes.

To summarize, you should <u>not</u> worry about the constant at first. Write an initial program without a constant, using loop tables or pseudocode to help you deduce the patterns in the output. After your figure looks correct at the default size, begin a second version with the constant. See Chapter 2's case study for a good example program.